ATTACHMENT - A

A study was performed to examine the effects of the memory/logic connection block flexibility on the overall routability and speed of a circuit implemented on an FPGA with embedded array.[11] It was found that the routability of circuits with memory is not strongly affected by a low memory/logic interconnect flexibility, especially for architectures with fewer than eight embedded arrays. For such architectures, the logic routing sources are flexible enough to compensate for the low flexibility in memory/logic interconnect. However, this becomes less true as the number of arrays (and hence the number of connections to memory) is increased.

3.1.3 Fine-grain (or cellular) FPGAs

The fine-grain architecture allows direct connection between the neighboring cells, enabling the users to combine cells to form compact local functions. An example of fine-grain FPGAs is the Motorola programmable array (MPA) family based on Pilkington architecture (see Sec. 3.3.5). A key feature of this approach is the Pilkington array, which is partitioned into 100 cell zones, each of which can be considered as a separate array with the port cells forming an interface between the zone interconnect and the global interconnect. The global interconnect joins individual zones together to form the larger array.

The interconnect structure is hierarchical so that, at the lowest level, the fast connections are used to join local cells to form macros. These can efficiently implement small functions such as complex combinatorial logic, counters, and comparators. These functions are then combined using the medium interconnect within the zone and the highest-level global interconnect throughout the array for signal routing and complete layout. The short range of local connections and their limited loading means they can be quite fast.

The Motorola MPA architecture also differs from other coarse-grained, channeled-array devices in functionality of the cell, which provides only a small set of two input functions. All cells have a basic AND function along with a subsidiary function such as an XOR or D-type flip-flop. In the channeled array architecture, such a cell would be very inefficient, with a high routing overhead between each level of logic. However, using a hierarchical routing structure, these small cells can be very efficiently combined into macros with negligible routing overhead. There are two basic advantages, as follows:

1. There is no redundancy, i.e., if a cell is used to perform simple function, there are no additional logic resources in the cell to be wasted.
2. The cell can be highly optimized, since the critical paths through the cell are limited in number.

The functions of the cells in the array are no longer homogeneous, but vary over a group of cells called a *tile*. These are repeated uniformly over the array. Thus, the fine-grained structure of this architecture is well suited to logic synthesis tools, and automatic design and layout.

3.2 Programming Technologies

A high-performance FPGA requires a programmable interconnect switch that occupies a small area and, at the same time, has a low parasitic resistance and capacitance. The other major attributes for programming technology consideration are series on-resistance of the programmed switch, its volatility and reprogrammability, and process complexity. Several different programming technologies are used to implement the programmable switches in FPGAs. The three most commonly used programming technologies are as follows:

- Antifuse switch of dielectric or amorphous silicon composition, which on electrical programming forms a low-resistance interconnect path
- SRAM-based technology in which the switch is a pass transistor controlled by the state of a RAM bit in a look-up table (LUT)
- EPROM-based technologies (either UVEPROM or EEPROM) in which the switch is a floating-gate transistor that can be turned off by injecting charge on to the floating gate

In all these cases, the programming switch element occupies a larger area and exhibits higher parasitic resistance-capacitance compared to a typical contact or via used in MPGAs. Therefore, the performance achievable by current generation of FPGAs is usually about an order of magnitude lower than that for the MPGAs manufactured using the same process technology.

The floating-gate EPROM technology (UVEPROM, EEPROM, and flash EEPROM) has been used in complex programmable logic devices (CPLDs) such as Altera, Lattice, and Xilinx families, discussed in Chap. 2. The floating-gate technology has also been used for Gatefield ProASIC family fine-grained FPGAs, which are based on a proprietary flash-EPROM controlled switch that is both nonvolatile and reprogrammable (see Sec. 3.3.8). However, for majority of FPGAs, the antifuse and SRAM are the dominant programming technologies. The

BEST AVAILABLE COPY

following sections discuss only these two technologies and their relative trade-offs.[11]

3.2.1 Antifuse programming technology

An antifuse is basically a two-terminal device with an unprogrammed state representing a very high resistance (several hundred megohms) between its terminals. On the application of a high voltage (from 11 V to 21 V), depending on the device and antifuse type used, the antifuse is "blown" to create a low-resistive, permanent link (or connection). The two most commonly used antifuse technologies are: (a) oxide-nitride-oxide (ONO) dielectric based and (b) amorphous silicon, or metal-to-metal antifuse structures. The programming of antifuses requires extra on-chip circuitry to deliver the high programming voltage (and relatively high current) through the pass transistors.

The dielectric antifuses consist of a layer of dielectric material placed between N' diffusion and polysilicon. This dielectric breaks down upon an application of sufficient voltage. Early dielectric antifuses used a single-layer oxide dielectric. A *programmable low impedance circuit element* (PLICE), which is a multilayer oxide-nitride-oxide (ONO) dielectric fuse, was developed for use in Actel FPGAs. An application of programming pulse (16 to 21 V) across the PLICE melts the dielectric, creating a conductive link of polycrystalline silicon between the electrodes. Figure 3.4*a* shows the cross section of PLICE antifuse structure. A thin layer of oxide is thermally grown on top of the N^+ surface, followed by a layer of low-pressure chemical vapor deposition (LPCVD) nitride and the reoxidized top oxide. The PLICE adds three masks to a conventional double-metal CMOS process.

The manufacturers of antifuse-based FPGAs provide their own programming algorithms as part of the device specification and application information. In all types of antifuses, the smaller the thickness of a given antifuse material, the lower the link resistance for given current, and higher the leakage current for unprogrammed state.[12] Also, the more current applied during programming, the lower the resistance of the link, and link resistances can be improved by appropriate programming waveforms and soaking times. Since the programming current amount has a large effect on link resistance, the programming circuits for antifuses need to supply high currents (e.g., 15 mA for Actel devices) for reliable configuration of high-performance interconnects.

In its simplest form, an antifuse array is a collection of vertical and horizontal wires with an antifuse at every crossing (or intersection), as shown in Fig. 3.4*b*.[13] Let us define V_{pp} as the programming voltage needed to program any fuse. Then, typically, the programming of an antifuse in the array requires three steps, as follows:

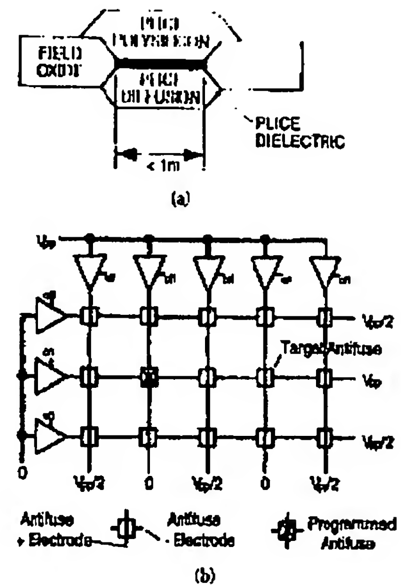

Figure 3.4 (a) Cross section of a PLICE antifuse structure (*from Ref. 15*) and (b) an array of antifuses (*from Ref. 13*).

1. Precharge all vertical/horizontal wires to $V_{pp}/2$.
2. Select the antifuse to be programmed (address) and drive the vertical (horizontal) wire connected to the target antifuse to 0 V, and select (address) and drive the horizontal (vertical) wire connected to the target antifuse to V_{pp}. This ensures that only the target antifuse is stressed to V_{pp}, and the remaining antifuses remain at 0 V or $V_{pp}/2$ (see Fig. 3.4*b*).
3. On antifuse breakdown, soak the antifuse with high current to stabilize the newly formed conductive link and to lower the antifuse resistance.

The programming voltage and current are generally controlled by analog circuits (programmers) external to the FPGA chip. The addressing requires independent horizontal and vertical decoding controls, and these schemes vary from one FPGA vendor to another. Actel FPGAs use an indirect addressing scheme to apply V_{pp} and 0 V to the target antifuse electrodes after precharge to $V_{pp}/2$.[14] For Actel antifuses, the programming algorithm specifies the peak programming voltage (V_{pp})

BEST AVAILABLE COPY

as 21 V, I_{max} (clamp current) = 15 mA, and number of soak pulses from 30 to 800. A confirmation that an antifuse has been programmed is made through monitoring the current on V_{pp} pin, which is typically <10 μA for an unprogrammed antifuse. Once an antifuse is programmed and an electrical connection is made between V_{pp} and ground, 3 to 15 mA current may be observed on V_{pp}. Once the antifuse is considered programmed and enters "soak" cycle, extra pulses are applied to the antifuse to achieve minimum resistance.

In contrast to Actel's indirect addressing scheme, QuickLogic uses direct addressing to program the antifuses. Since no pass transistors are used, QuickLogic uses a dedicated driver plus shift register bit(s) per wire segment. The number of segments addressed is equal to the number of shift register/driver cells.

An alternative to dielectric based antifuse has been the development of amorphous (noncrystalline) silicon antifuse technology. This is based on the principle that a layer of amorphous silicon placed between two metal layers undergoes a phase charge when a current is passed through it, and it becomes conductive. In its unprogrammed state, the amorphous silicon is an insulator with resistance greater than 1 GΩ. The user can program an antifuse link by applying relatively high current (roughly 20 mA) signals that effectively convert the insulating amorphous silicon into conductive polysilicon link.

An example of amorphous silicon antifuse application is QuickLogic pASIC FPGA families, based on ViaLink, which provides a low-resistance, low-capacitance programmable connection directly from one metal layer to another. The ViaLink element is formed by depositing a very high resistance (>1 GΩ) of amorphous silicon above a tungsten via plug that would otherwise bridge the insulation between the two metal layers. On the application of a programming voltage to a selected via, a direct metal-to-metal link is formed with typical low-resistance values of less than 50 Ω. In a 0.35 μm CMOS process, the size of a ViaLink is roughly 1 μm^2, which is orders of magnitude smaller than the active elements, resulting in low capacitive loading [<1 fF (femtofarad, 10E^{-15} farads)] and improved speed performance. Figure 3.5 shows (a) a three-layer metal ViaLink structure, (b) an unprogrammed ViaLink element, and (c) a programmed ViaLink element.[18]

3.2.2 SRAM-based programming technology

SRAM-based FPGA architecture was discussed in Sec. 3.1.2. The SRAM programming technology uses static RAM cells to control pass gates or multiplexers. The speed of an FPGA is a measure of the delay required to implement a function and to propagate signals to the neighboring functions. The speed of the logic part of the block is the

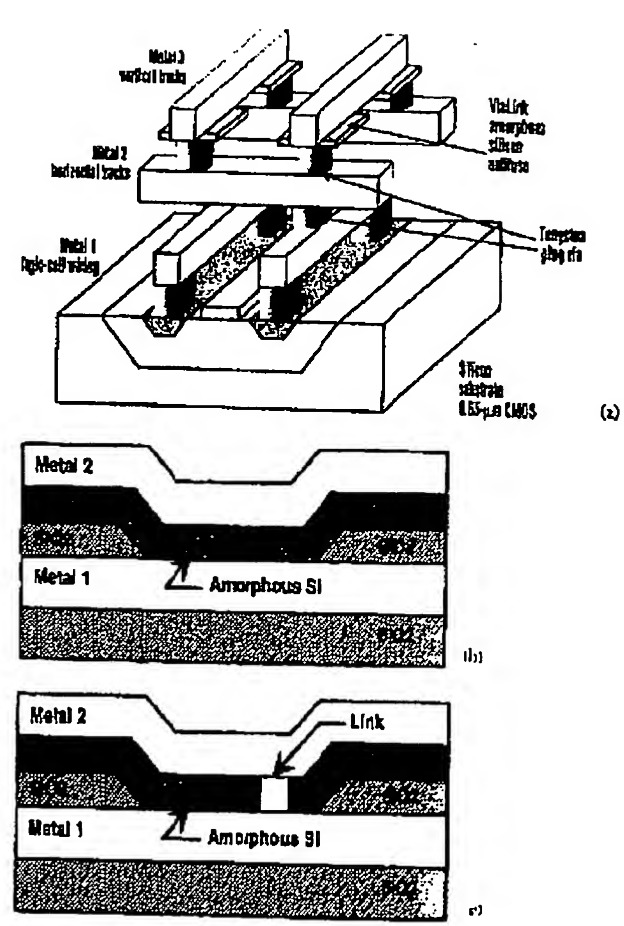

Figure 3.5 (a) Three-layer metal ViaLink structure, (b) unprogrammed ViaLink element, and (c) programmed ViaLink element (from Ref. 18).

sum of delays from the input selection multiplexers, the look-up table, and the output drivers. The speed and density trade-offs are based on the user's design requirements. An architecture with more programmable interconnect points (PIPs) provides more routing flexibility, but each PIP adds to the size of FPGA and the routing delay.

Therefore, the FPGA size and performance trade-offs need to take into consideration not only the logic elements but also the interconnect segments, PIPs, and the multiplexers required to connect the elements. The LUT width is also an important factor. For example, a four-

input LUT has sixteen memory cells and a three-input LUT has only eight. If the logic being implemented in a four-input LUT breaks naturally into three input gates, half of every four-input LUT area will be wasted, leading to an inefficient design implementation. A study was performed to investigate a range of LUTs and their effect on overall logic block and routing interconnect area.[1] The results showed that a three-input or four-input LUT provided the best density for a wide range of programming cell sizes. The larger LUTs were preferred for high-speed architecture.

3.2.3 Antifuse versus SRAM-based technology trade-offs

A review of general FPGA architectures and programming technologies shows that both the antifuse- and SRAM-based devices have relative advantages as well as disadvantages. The selection of a particular programming technology for an FPGA is primarily based on the user's application requirements and involves trade-offs in areas such as design logic utilization capacity, performance, and in-system programmability features. There are proponents and critics for both of these approaches. The following are some of the relative advantages and disadvantages of each technology:

- Antifuse technology is inherently faster than SRAM-based technology because of lower interconnect resistance and capacitance values. Antifuse is much smaller than a transistor-RAM cell combination and has much lower "on" resistance and capacitance than a minimum-sized transistor. As an example, for 0.8 μm CMOS technology, the impedance of pass transistor in SRAM-based FPGA is about 250 Ω, as compared to about 50 Ω resistance for QuickLogic ViaLink metal-to-metal antifuses. As for the capacitance values, it is only 1 fF for unprogrammed ViaLink element compared to 50 fF for pass gate transistor capacitance. Since the net delays are determined by RC time constraints, the lower resistance and capacitance of the antifuse networks provides significant smaller delays than the SRAM-based pass gate transistor networks.
- For equivalent silicon area per gate, antifuse FPGAs provide more flexibility and are easier to route compared to the SRAM-based FPGAs. SRAM pass-gate interconnects require at least four transistors for the flip-flop and one transistor for the pass gate. This structure implies limited use of interconnect resources and forces macrocell architecture grain size to be coarse.
- A disadvantage of antifuse-based technology is that it requires more process layers and mask steps than the SRAM-based FPGAs and use on chip, high voltage programming transistors. SRAM-based FPGAs are easier to migrate to the next process generation and, unlike antifuse based devices, the performance of both logic elements and programmable interconnect benefit from scaling to smaller geometries. However, the proponents of antifuse-based FPGAs claim that the smaller physical size of antifuse results in less expensive die, and the area required to hold a pass gate transistor and associated memory cell is certainly larger than the area required for an antifuse.
- A survey of FPGA market shows that the SRAM-based FPGAs offer higher capacity than the antifuse-based devices. Some SRAM-based FPGAs feature a power-down mode in which the power is supplied only to the memory cells that hold the configuration program.
- One of the major advantages of SRAM-based FPGAs is the flexibility offered by their in-system reconfigurability, which makes them a good candidate technology for prototype development. This feature can significantly ease the debugging process and reduce overall system design and development costs. In contrast, the antifuse technologies are one-time programmable (OTP) configurations.
- Another advantage often cited for the SRAM-based FPGAs reprogrammability feature is that the parts can be fully tested at the factory, as compared to the antifuse based devices, which are tested as "blanks," since the antifuses are user programmable per design requirements.
- A disadvantage of the SRAM-based FPGAs is that the programming is volatile, i.e., when the power is turned off, the FPGA loses its configuration program information. Therefore, the SRAM-based FPGAs must be reprogrammed each time power is applied, as part of turn-on initialization sequence. This initialization sequence requires an external memory for permanent (nonvolatile) storage of the program.

3.3 FPGA Vendor Families and Development Tools

3.3.1 Actel FPGAs

Actel FPGAs are based on the channeled array segmented architecture and PLICE antifuse technology discussed in Secs. 3.1 and 3.2, respectively. Actel FPGA offerings consist of following device families:

- ACT 1 family, with up to 2000 gate array gates (6000 PLD equivalent gates)

BEST AVAILABLE COPY